Stateful Identity-Based Encryption Scheme: Faster Encryption and Decryption

[Extended Abstract]

Le Trieu Phong[*]
Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku
Tokyo, 152-8550, Japan

Hiroto Matsuoka[†]
Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku
Tokyo, 152-8550, Japan

Wakaha Ogata[‡]
Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku
Tokyo, 152-8550, Japan

ABSTRACT

In this paper, we first present the notion of stateful identity-based encryption (IBE) and then extend standard security definitions for IBE to the stateful setting. After that, we demonstrate a concrete stateful IBE scheme, whose security meets the strongest definition in the setting in random oracle model, and whose encryption and decryption are very efficient, compared to existing IBEs: one pairing each for encryption and decryption.

Keywords

IBE, stateful encryption, provable security, ROM

1. INTRODUCTION

Motivation. Recently, Bellare et al. [2] showed how to encrypt with just one 160-bit exponentiation in some public-key encryption schemes. This is a dramatic improvement with respect to encryption speed, since most previous schemes require more exponentiations. For example, the Kurosawa-Desmedt variant [11] of the Cramer-Shoup public-key encryption schemes [8], which is considered one of the most efficient schemes so far, needs three exponentiations (or more) for encryption. The reason why the public-key encryption schemes of Bellare et al. [2] achieve such encryption efficiency is from the idea of making them stateful. Namely, some randomness and exponentiations are kept unchanged throughout many encryptions as state, so that the encryption algorithms do not have to compute the values again and again any more.

Bellare et al. [2] have focused on the case of public-key encryption (PKE) schemes. As with the usual (i.e. non-stateful) PKE ones, the management of the public keys may cause obstacles for deployment of stateful PKE schemes in the real world, since each user must be securely linked to his public key. To avoid the obstacles related to the public-key management, in this paper, we show how to use the idea of stateful encryption in another important asymmetric setting: Identity-Based Encryption (IBE). In particular, we first define the model of security for stateful IBE scheme and then present a concrete stateful IBE scheme whose security meets the (strongest) definition in the model, while its encryption and decryption algorithms are also extremely efficient. In particular, we show how to encrypt securely with just one pairing in identity-based setting. Let us now look at the above statements in some more details.

The Stateful Model and Security Definition for Stateful IBE Schemes. Briefly, there are two security issues to be considered. Roughly speaking, the first issue, which is the same as in definitions in literature, is the coalition of users should not compromise the privacy of any other user. As with the definitions in the literature[4], [7], this issue is addressed by giving the adversary against stateful IBE schemes a private key extraction oracle.

However, using a stateful IBE scheme, a sender maintains state information which can be used throughout his/her many encryptions (e.g., in a work day.) This use of state causes some path for attack not covered by the classical definitions of security for (non-stateful) IBE schemes (e.g., IND-ID-CCA [4]). This fact leads us to the second issue below.

The second issue, which is raised only in stateful model, is that encrypted messages can be dependent, since they might depend on the same or related state (which is unknown to the adversary), and hence the ability of seeing numerous ciphertexts (from one sender to many receivers, namely identities) might lead to the ability of compromising the privacy of the sender. We address this path of attack by giving the adversary an encryption oracle, from which it can adaptively obtain ciphertexts computed under the sender's state, and any identity ID, any message of its choice.

Definitions captured the both issues are provided in Section 2. We also make some comparison and contrast with Bellare et al.'s definitions, which concentrate on public-key encryption schemes. As will become evident, stateful en-

[*]ltphong@crypt.ss.titech.ac.jp
[†]hiroto@crypt.ss.titech.ac.jp
[‡]wakaha@mot.titech.ac.jp

ASIACCS '08, March 18-20, Tokyo, Japan

cryption apparently fits IBEs more than PKEs. Below, when saying a stateful IBE scheme is IND-ID-CCA secure, we mean that it meets the corresponding definition in Section 2.

A Concrete Stateful IBE Scheme: SIdent. This is a stateful variant of the Boneh-Franklin IBE scheme called BasicIdent [4], which was proven to be IND-ID-CPA secure in non-stateful setting. On the contrary, we show in this paper that the variant enjoys being IND-ID-CCA secure in the stateful setting, while being able to save two exponentiations in encryption. Indeed, SIdent requires (at most) one pairing for encryption, while its original version needs two exponentiations and one pairing. Costs for decryption in the both scheme are the same, which is one pairing. Note that since pairing computation is more expensive than that of exponentiation in general, it would be more desirable to reduce the number of pairings. However, we comment that the use of pairing is tightly related to the BDH assumption (and related ones), so one pairing in the encryption and decryption of SIdent would be unavoidable.

At the same CCA-security level, we would like to compare the proposed SIdent with the fully secure version of BasicIdent, which is called FullIdent in [4]. With respect to computation cost, SIdent is much more efficient, beating FullIdent at a margin of two exponentiations in encryption and one exponentiation in decryption. (The latter requires two exponentiations (also one pairing) for encryption and one exponentiation (also one pairing) for decryption.)

With respect to security assumption, the security of the scheme SIdent is based on the Gap Bilinear Diffie-Hellman (Gap-BDH) assumption, while that of FullIdent is relied on the BDH assumption, which is potentially weaker. However, the factor loss in our reduction for SIdent, approximately q_H, is tighter than that in the latter, approximately q_H^3[1], where q_H is the bound for the number of hash queries (to any hash function in the schemes). This result has its own interest, since a tighter reduction from a given assumption can be preferred to a loose reduction from a potentially weaker assumption.

Our Contributions. In summary of what have been discussed above, the main contributions of this paper are: extending the security notion of usual IBE schemes to the case of stateful IBE schemes, and showing that the security is achievable and meaningful by demonstrating the concrete scheme SIdent whose security meets the IND-ID-CCA definition in the stateful setting. Furthermore, the scheme SIdent is extremely efficient: only one pairing each for encryption and decryption.

Random Oracle Model. For schemes in the random oracle (RO) model [5], any of the constituent algorithms may have access to a RO, and in the security game, the adversary has access to it as well. We will use this model in the security analysis of SIdent.

2. DEFINITIONS

In this section, we formalize the definition of stateful IBE scheme, and security notions for it. We then compare and contrast one of the security definitions with its counterpart in public-key setting.

Stateful IBE Scheme. A stateful identity-based encryption scheme StIBE = (Setup, Extract, NwSt, Enc, Dec) is specified by five algorithms (all possibly randomized except the last).

Setup: takes a security parameter κ and returns system parameters params, which is publicly known, and master key msk, which is kept secret.

Extract: takes as input params, msk, an identity ID, and returns a private key SK_{ID}.

NwSt: (New State) takes as input params and outputs the system state st, which will be used for encryption.

Enc: takes params, an identity ID, a message m, and a system state st, and returns a ciphertext C and an update system state st (may be the same with the previous one or not.)

Dec: takes as input a ciphertext C, and identity ID and a private key SK_{ID}, and returns a message m (may be reject).

As we can see, there are some differences between a stateful IBE scheme with a (non-stateful) IBE scheme, which are in NwSt, and Enc. NwSt creating system state st, which is used by Enc, is the basic algorithm distinguishing a stateful IBE scheme from a non-stateful one. The algorithm Enc also returns an update system state beside a ciphertext. In all concrete stateful IBE schemes later in this paper, Enc does not modify the state and just returns the same state.

IND-ID-CCA Security for Stateful IBE. To define IND-ID-CCA security of a stateful IBE scheme StIBE = (Setup, Extract, NwSt, Enc, Dec), we consider an adversary A played with the following (ind-id-cca) game:

- Setup: The algorithm Setup is run with a security parameter κ generating system parameters params and master key msk. params is given to A, while msk is kept secret.

- State Generation: The game runs the algorithm NwSt with input params to generate a state st, which will be used for encryption.

- Phase 1: The adversary A adaptively issues queries $q_1, \dots, q_m$ of the following types:

  - Extraction query ID_i. The game responds by running algorithm Extract to generate the private key SK_{ID_i}, and returns this private key to A.
  - Decryption query (ID_i, C_i). The game returns Dec(SK_{ID_i}, C_i), where SK_{ID_i} is obtained from Extract.
  - Encryption query (ID_i, m_i). The game runs the algorithm Enc to encrypt m_i using identity ID_i under the encryption state st. The output of Enc is a ciphertext and a state, but only the ciphertext is given to A.

[1] This value is borrowed from [10] (full version, p. 9). That work showed a flaw in the change from BasicIdent to FullIdent via the Fujisaki-Okamoto transformation [9]. Our work does not use the transformation, so it does not fall into the mistake.

- Challenge: The adversary outputs two equal-length plaintexts m_0, m_1 and an identity ID^* on which it wants to be challenged. The only constraint is that ID^* did not appear in any private key extraction query in Phase 1. The game in turn picks a random bit $b \in \{0,1\}$ and computes $(C^*,\ st) \xleftarrow{\$}$ Enc(params, ID^*, m_b, st), and only C^* is returned to A.
- Phase 2: The adversary may adaptively issue more queries $q_{m+1}, \ldots, q_n$ of the following types:
  - Extraction query $ID_i (\neq ID^*)$. The game responds as in Phase 1.
  - Decryption query $(ID_i, C_i) \neq (ID^*, C^*)$. The game responds as in Phase 1.
  - Encryption query (ID_i, m_i). The game responds as in Phase 1.
- Guess: Finally, A outputs a bit b' as its guess of b.

Define the ind-id-cca-advantage of A:

$$\text{Adv}_{stIBE}^{ind-id-cca}(A) = |\Pr[b' = b] - \frac{1}{2}|.$$

We say that stIBE is IND-ID-CCA secure if the advantage of any polynomial time adversary A is negligible in the security parameter κ.

The above notion follows the notion in the literature [4], except that the adversary A now can have its own messages encrypted with its maliciously-chosen identities, under a fixed (and unknown to A) encryption state. This additional feature aims to capture the adversary ability of observing encrypted messages which are dependent under the same or related system state, as discussed in the second issue mentioned in Section 1.

IND-ID-CPA/ IND-sID-CPA/ IND-sID-CCA. The above definition can be modified to obtain more kinds of security in a usual manner. Briefly, in an IND-ID-CPA game, the adversary A is not allowed to submit (ID_i, C_i) for decryption as above. IND-sID-CPA game is the same as the IND-ID-CPA one, except that A must output the target identity ID^* at the beginning of the game, before Setup is run. Allowing the decryption query (ID_i, C_i) in the IND-sID-CPA game gives us IND-sID-CCA game. Note that the notion of selective-ID (sID) is first introduced by Canetti et al. [7].

Comparison and Contrast with Public-Key Setting. In the first place, security definitions for stateful IBEs are more natural, compared to its counterparts for stateful PKE. In details,

- In public-key setting [2], an adversary is challenged on a random,"honest" public key. The adversary is able to create malicious public-keys via an oracle, and can have messages encrypted under these public-keys (and the honest public-key also) via another oracle. These correspond to the encryption oracle in the above notion, although in ID-based setting, the adversary is challenged on a target identity of its own choice. The fact that all identities (except the challenge one) might be malicious simplifies the security definition for stateful IBE schemes, since we do not need the oracle creating malicious public-key any more.
- Also, in [2], Bellare et al. separate the unknown secret key (USK) model from the known secret key (KSK) model. Briefly, the KSK model means the adversary has to possess a corresponding secret key when registering a malicious public-key, while in the USK model it does not have to. They needed the KSK model for the security proof of the stateful variant of the Kurosawa-Desmedt encryption scheme [11]. In contrast, in ID-based setting, the adversary knows the corresponding private key SK_{ID} of all identity ID $(\neq ID^*)$ via the key extraction oracle, so the separation between the USK and KSK model is not necessary any more.

In the second place, the fact that the state of stateful IBEs relies on system-wide parameters params, which are used by all identites for encryption and decryption, appears to open more ways to reduce encryption cost, compared to stateful PKEs where each user needs a different public key. We use this fact (together with an algebraic property of bilinear map) to get rid off all exponentiations in SIdent, as will be seen later.

In short, we gain more when using states in IBEs. Although formalized later and can be viewed as an adaptation from PKEs to IBEs, stateful encryption apparently fits IBEs more than PKEs.

An Extended Security Model and Equivalence with the Basic One. The above notion of security is a simplified one, considering only one sender with a fixed encryption state. Like [2], one can consider an extended model where there are many senders with many states, each of which may be reset or compromised. Specifically, the adversary can has access to three more oracles for creating senders with their states, reseting a sender state, and revealing a sender state with a natural restriction that the challenge state cannot be revealed or reset. This extended security notion implies forward-security of the system, namely it retains security even if previous states are compromised. We note that the simplified notion is equivalent to the extended one up to a polynomial loss factor. The equivalence proof is via a standard guessing argument, and is almost identical to its counterpart in [2], so we omit it. We therefore stick to the simplified security notion in the rest of this paper.

3. THE SIDENT SCHEME

This section is devoted to presenting the proposed SIdent scheme. We first describe its building blocks, and then the scheme, and finally analyze its security.

Building Blocks. We first describe the building blocks used and the assumptions about them.

- A BDH parameter generator $\mathcal{G}$ whose input is security parameter κ and output is a quadruple $(q, G_1, G_2, \hat{e})$, where q is a prime, G_1 and G_2 are cyclic groups of order q, and $\hat{e}$ is a pairing from $G_1 \times G_1$ to G_2. Let P be a random generator of G_1.
  - The BDH problem is: given (P, aP, bP, cP) for $a, b, c \xleftarrow{\$} Z_q$, compute $\hat{e}(P,P)^{abc}$.
  - The decisional BDH (DBDH) assumption is: distinguish the tuples of the form (P, aP, bP, cP, $\hat{e}(P,P)^{abc}$) from those of the form (P, aP, bP, cP,

Z), where $a, b, c \xleftarrow{\$} Z_q$ and $Z \xleftarrow{\$} G_2$. A DBDH oracle $\mathcal{O}_{dbdh}$ is an algorithm which, given (P, aP, bP, cP, Z) as input, outputs 1 if $Z = \hat{e}(P, P)^{abc}$ and outputs 0 otherwise.

- The Gap-BDH problem is: solve the BDH problem with the help of the DBDH oracle. We assume the Gap-BDH problem is hard with respect to $\mathcal{G}$, captured by defining the gap-bdh-advantage of an adversary $A_{\mathcal{G}}$ as

$$Adv_{\mathcal{G}}^{gap-bdh}(A_{\mathcal{G}}) = \Pr[Z = \hat{e}(P, P)^{abc}]$$

in the following experiment: $P \xleftarrow{\$} G_1{}^*; a,b,c \xleftarrow{\$} Z_q$; $Z \leftarrow A_{\mathcal{G}}^{\mathcal{O}_{dbdh}}(P, aP, bP, cP)$.

- A symmetric encryption scheme $SE = (SEnc, SDec)$ with key-length k. The scheme is assumed IND-CCA secure, captured by defining the ind-cca-advantage of an adversary A_{SE} as

$$\mathrm{Adv}_{SE}^{ind-cca}(A_{SE}) = |\Pr[d = c] - \frac{1}{2}|$$

in the following experiment: $K \xleftarrow{\$} \{0, 1\}^k$; $c \xleftarrow{\$} \{0, 1\}$; $d \xleftarrow{\$} A_{SE}^{\mathcal{O}}$ where $\mathcal{O}$ is one of the following: $SEnc(K, \cdot)$, $LOR(K, \cdot, \cdot, c)$, $SDec(K, \cdot)$. Here, $LOR(K, m_0, m_1, c)$ returns $C_s^* \xleftarrow{\$} SEnc(K, m_c)$. A_{SE} is allowed only one query to this left-or-right oracle, consisting of a pair of equal-length messages, and it is not allowed to query $SDec(K, \cdot)$ on C_s^*.

We note that the security for SE is easily and cheaply obtained via encrypt-then-mac generic composition [3].

Scheme and Security. The scheme SIdent = (Setup, Extract, NwSt, Enc, Dec) is described as follows.

Setup: Given a security parameter $\kappa \in \mathbb{Z}^+$, the algorithm works as follows: (1) Run the BDH generator on input κ to generate a prime q, two groups G_1 and G_2 of order q, and a pairing $\hat{e} : G_1 \times G_1 \rightarrow G_2$. Also choose a random generator $P \in G_1$. (2) Pick $s \xleftarrow{\$} \mathbb{Z}_q^*$ and $P_{pub} \leftarrow sP$. (3) Choose two cryptographic hash function $H_1 : \{0, 1\}^n \rightarrow G_1^*$, and $H_2 : G_2 \rightarrow \{0, 1\}^k$, where n is identity length and k is the key length of the symmetric encryption scheme.

The public parameters $params := (q$, G_1, G_2, $\hat{e}$,P, P_{pub}, n, k, H_1, $H_2)$ and master secrete key $msk := s$.

Extract: Given the master secret key s and an identity $ID \in \{0, 1\}^n$, the algorithm computes $Q_{ID} \leftarrow H_1(ID)$, $SK_{ID} \leftarrow sQ_{ID}$, and outputs SK_{ID} as the private key for identity ID.

NwSt: Given the public parameter $params$, the algorithm picks $r \xleftarrow{\$} Z_q^*$ and outputs (rP, rP_{pub}) as the state of the scheme.

Enc: Given $params$, an identity ID, a message $m \in \{0, 1\}^*$, and an encryption state st, the algorithm does as follows: (1) parses st as (R, R'), (2) computes $Q_{ID} \leftarrow H_1(ID)$, (3) sets $K \leftarrow H_2(Q_{ID}, R, \hat{e}(Q_{ID}, R'))$, and (4) computes $C \xleftarrow{\$}$ (R, $SEnc_K(m)$). The algorithm returns the ciphertext C and the unmodified state st.

Dec: To decrypt ciphertext C under identity ID and its corresponding private key SK_{ID} does the following: (1) parse C as (S, C_s), (2) compute $Q_{ID} \leftarrow H_1(ID)$ and $K \leftarrow H_2(Q_{ID}, S, \hat{e}(SK_{ID}, S))$, and (3) output $SDec_K(C_s)$ (may be reject).

This completes the description of SIdent. Its correctness comes from the fact that the symmetric key K is the same in both encryption and decryption algorithms when everything is computed as above, since $\hat{e}(Q_{ID}, R') = \hat{e}(sQ_{ID}, rP) = \hat{e}(SK_{ID}, R)$.

It is worth noting that in NwSt, r is not output. This is natural as done in most encryption schemes, and is in sharp contrast with all the PKE schemes in [2]. The reason we can omit r comes from the advantage of having the state rely on system-wide parameters $params$. Indeed, the value P_{pub} is in $params$, so we can use rP_{pub} as a part of the state. This will be impossible in a PKE related to SIdent below, since P_{pub} will become a part of recipient's public key.

It appears that the encryption algorithm Enc is deterministic, but it does not. The randomness for the encryption comes from the (IND-CCA secure) symmetric encryption algorithm $SEnc$, which is an important component leads to the following result.

Theorem 1. Consider SIdent associated to BDH generator $\mathcal{G}$ and symmetric encryption scheme SE as above, and suppose the hash functions H_1, H_2 are random oracles. Then SIdent is IND-sID-CCA secure. More precisely, let A be an IND-sID-CCA adversary against SIdent, then there are Gap-BDH adversary $A_{\mathcal{G}}$ with respect to $\mathcal{G}$, and adversary A_{SE} against SE, whose running times are essentially the same as that of A, such that

$$Adv_{\mathrm{SIdent}}^{ind-sid-cca}(A) \leq Adv_{\mathcal{G}}^{gap-bdh}(A_{\mathcal{G}}) + Adv_{SE}^{ind-cca}(A_{SE}).$$

The proof ideas of the theorem are as follows: we first show that an attack against SIdent is reduced to an attack against a stateful PKE scheme (defined below), by considering H_1 as a random oracle. Then by modeling H_2 as a random oracle, we reduce the attack against that stateful PKE scheme to solving the Gap-BDH problem and attacking the underlying symmetric encryption scheme. Although the roadmap of the proof is the same as Boneh-Franklin [4], the proof details are completely different, which we will see later.

Proof. Recall that in the IND-sID-CCA game, A has to commit a target identity ID^* at the beginning of the game, before Setup is run. And in stateful setting, A further can ask queries of the form (ID, m) to an encryption oracle, as defined in Section 2. We first want to make the stateful IND-sID-CCA game simpler by using a nice feature of SIdent, so that all identities used in the encryption oracle are only the identity ID^*. Indeed, suppose A submits (ID, m), where $ID \neq ID^*$, as an encryption query. In this case, A also can know SK_{ID} by the key extraction oracle, so that it is able to encrypt m itself as follows: $K \leftarrow H_2(H_1(ID), R, \hat{e}(SK_{ID}, R))$, $C \leftarrow (R, SEnc_K(m))$, where R is the first part of the encryption state. The point now is how A obtains R, but this is easy since A just makes an encryption query (not a challenge query) like (ID^*, "$GetR$") in order to get R. The reasoning thus far enables us to assume without weakening adversary's ability that A only makes queries of the form $(ID^*, \cdot)$ to the encryption oracle. The following Lemma 1 will take advantage of this fact, and the theorem follows directly from Lemma 1 and Lemma 2 below. □

Before going further stating and proving Lemmas 1 and 2, we first define a stateful PKE scheme, related to the

IBE SIdent, called $SPub$ = ($SPub.Setup$, $SPub.Keygen$, $SPub.NwSt$, $SPub.Enc$, $SPub.Dec$), using the same building blocks as the IBE SIdent. The components of $SPub$ are described as follows.

The algorithm $SPub.Setup$ takes a parameter κ as input, and returns $sp = ((q, G_1, G_2, \hat{e}), P, H_2)$, which respectively consists of the output of the BDH generator $\mathcal{G}$, a random generator of group G_1, and a hash function from G_2 to $\{0,1\}^k$. With that sp as input, $SPub.Keygen$ chooses $s \xleftarrow{\$} \mathbb{Z}_q$, $Q \xleftarrow{\$} G_1$, and sets $P_{pub} \leftarrow sP$, and finally returns $pk = (P_{pub}, Q)$ as the public-key, and $sk = sQ$ as the secret key. With the parameter sp as input, the algorithm $SPub.NwSt$ chooses $r \xleftarrow{\$} \mathbb{Z}_q$, sets $R \leftarrow rP$, and returns $st = (r, R)$ as encryption state. $SPub.Enc$ takes as input the parameter sp, public key $pk = (P_{pub}, Q)$, a message m, and the encryption state $st = (r, R)$; it then computes $K \leftarrow H_2(Q, R, \hat{e}(Q, P_{pub})^r)$, lets the ciphertext be $C \xleftarrow{\$} (R, SEnc_K(m))$ and finally returns both C and the unchanged state st. $SPub.Dec$ takes sp, pk, $sk = sQ$ and a ciphertext $C = (S, C_s)$ as input, and computes symmetric $K \leftarrow H_2(Q, S, \hat{e}(sk, S))$ and outputs $SDec_K(C_s)$. This completes the description of stateful PKE $SPub$, whose correctness is checked similarly as the stateful IBE SIdent.

The purpose of changing SIdent to $SPub$ is to show that the key extraction queries do not help the adversary against SIdent. Indeed, we will show that any attack against SIdent is reduced to an attack against $SPub$. (The attack against $SPub$ will be compactly presented in Lemma 1 below.) The idea of changing an IBE scheme to a PKE scheme was used to prove the security of (non-stateful) BasicIdent [4], but the proof of Boneh-Franklin and the proof here are quite different, and we will comment about the differences after the following lemma:

Lemma 1. Let $H_1 : \{0,1\}^n \rightarrow G_1^*$ be a random oracle, and A is an ind-sid-cca adversary against SIdent. Then there exists an adversary B against $SPub$, whose running time is essentially the same as that of A, such that

$$Adv_{\mathrm{SIdent}}^{ind-sid-cca}(A) = Adv_{SPub}^{ind-cca}(B),$$

where the right-hand side is defined as follows

$$Adv_{SPub}^{ind-cca}(B) = |\Pr[b' = b] - \frac{1}{2}|$$

in the following experiment: $b \xleftarrow{\$} \{0,1\}$; $b' \leftarrow B^{\text{in game G}}$; where the game G plays with algorithm B as follows: given parameter κ, the game first sets up: $sp \xleftarrow{\$} SPub.Setup(\kappa)$; $(pk, sk) \xleftarrow{\$} SPub.Keygen(sp)$; $st \xleftarrow{\$} SPub.NwSt(sp)$. The game then gives B the parameter κ, sp, and public key pk. B in turn has access to oracles $SPub.Enc(sp,\ pk,\ \cdot,\ st)$[2], $LOR(sp,\ pk,\ \cdot,\ \cdot,\ b,\ st)$ [3], $SPub.Dec(sp,\ sk,\ \cdot)$[4]. B can query the first and third oracles many times, but just one time to the second left-or-right LOR oracle. The first and second oracle outputs are a ciphertext and a state, but only the ciphertext is given to B. In addition, B cannot submit the output of LOR oracle to the third decryption oracle.[5]

The proof of this lemma is quite simple and is in Appendix A. The reduction in Lemma 1 is tight, as opposed to its counterpart in [4]. The point is, taking advantage of the selective-ID game, the adversary B is able to simulate the environment for A in a usual manner, avoiding Coron's technique [6] which made the reduction loose in [4].

Thanks to the fact that hash function H_2 can be seen as a random oracle, we have the following lemma.

Lemma 2. Consider the adversary B playing with game G as defined in Lemma 1, and suppose that H_2 is a random oracle. Then

$$Adv_{SPub}^{ind-cca}(B) \leq Adv_{\mathcal{G}}^{gap-bdh}(A_{\mathcal{G}}) + Adv_{SE}^{ind-cca}(A_{SE}),$$

where $A_{\mathcal{G}}$, A_{SE} are respectively the Gap-BDH adversary, and the adversary against symmetric encryption scheme SE.

The proof of this lemma, which is a bit intricate, is in Appendix B. The crux of it is as follows. Let $K^* = H_2(Q,$ $R, \hat{e}(Q,\ P_{pub})^r)$, where $R = rP$, be the (same) symmetric key in the oracles $SPub.Enc(sp, pk, \cdot, st)$ and $LOR(sp,\ pk,$ $\cdot,\ \cdot,\ b,\ st)$. We show that K^* is indistinguishable from a random key, based on the Gap-BDH assumption. With the random key, the security of $SPub$ then comes from the security of the symmetric encryption scheme SE.

4. FROM SELECTIVE-ID TO FULL SECURITY

Theorem 1 gives us an IND-sID-CCA secure stateful IBE scheme, but our goal is an IND-ID-CCA secure one. Results about transformation from selective-identity security to full security for non-stateful schemes are known in the literature [1]. To reach the goal without making unsubstantiated claims, we present in this section a generic way to transform an IND-sID-CCA secure stateful scheme to an IND-ID-CCA one. We first need some notations. Let Π is a stateful IBE scheme where all identities are bit strings of length n, and H be a hash function from $\{0,1\}^*$ to $\{0,1\}^n$. We denote by Π_H the stateful IBE scheme derived from Π by first hashing an arbitrary length identity, and then using Π with the hash string. Note that by this method, all identities in Π_H are of arbitrary length.

We say that a stateful IBE is $(t, q_{extr}, q_e, q_d, \epsilon)$ IND-sID-CCA secure (respectively, IND-ID-CCA secure) if no adversary with time complexity t, asking q_{extr} key extraction queries, q_e encryption queries, and q_d decryption queries has advantage (larger than) ϵ in the IND-sID-CCA game (respectively, IND-ID-CCA game). The security of Π_H is ensured by the following theorem.

Theorem 2. Model H as a random oracle. If Π is $(t,\ q_{extr},$ $q_e,\ q_d,\ \epsilon)$ IND-sID-CCA secure, then Π_H is $(t,\ q_{extr},\ q_e,\ q_d,$ $\epsilon')$ IND-ID-CCA secure with

$$\epsilon' = \epsilon(q_H + q_e)(1 - \frac{q_{extr} + q_d}{2^n})^{-1} \simeq \epsilon(q_H + q_e),$$

where q_H is the number of calls to H.

[2] who receives a message m.

[3] who receives two equal-length messages m_0, m_1 and then encrypts m_b.

[4] who receives a ciphertext C.

[5] This notion is similar to the definition of IND-CCA for stateful PKE schemes [2], except that the ability of adversary B is restricted: B can only gain encrypted messages under one challenge public-key pk, rather than its maliciously-chosen public keys as in [2]. Both $SPub$ and this notion just serve as a “bridge" for the proof of Theorem 1.

The proof of this theorem is in Appendix C.

Combining Theorems 1 and 2 lets us conclude that the scheme SIdent_H is fully secure under the Gap-BDH assumption, as stated in the following theorem.

Theorem 3. The notions are as in Theorems 1 and 2. We have

$$\begin{aligned} Adv^{ind-id-cca}_{\mathrm{SIdent}_H}(A) &\lessapprox (q_H + q_e) Adv^{gap-bdh}_{\mathcal{G}}(A_{\mathcal{G}}) \\ &+ (q_H + q_e) Adv^{ind-cca}_{SE}(A_{SE}). \end{aligned}$$

Remark that SIdent_H deals with identities of arbitrary length. Indeed, in SIdent_H, an identity $ID \in \{0,1\}^*$ is first hashed to $\{0,1\}^n$ by H, and then $H(ID)$ is processed by $H_1 : \{0,1\}^n \to G_1^*$. The whole process can be seen as $H_1 \circ H : \{0,1\}^* \to G_1^*$, which can be seen as one hash function. From the light of Theorem 3, all words SIdent in Section 1 should be precisely replaced by SIdent_H, since SIdent is not exactly fully secure. We however use SIdent for discussion in Section 1, for the sake of simplicity, and because the both schemes are the same except a hash function for identity.

Note that in practice, q_e, the number of oracle queries to the encryption oracle which uses the same state for encrypting is not large, compared to the number of hash queries q_H. We can thus assume that $q_e << q_H$, and hence the factor loss in the security reduction in Theorem 3 can be approximately considered as q_H. This value was used in the comparison with FullIdent in Section 1.

5. CONCLUSION

We have examined the definition of stateful IBE scheme, its corresponding security notions, and a concrete stateful IBE SIdent, which is IND-ID-CCA secure with tighter security reduction to the Gap-BDH assumption, and which is efficient in both encryption and decryption. Furthermore, it produces very short ciphetexts.

The scheme SIdent is considered in ROM. It is interesting to ask whether there are stateful IBE schemes whose securities are proven in standard model (i.e., not ROM).

6. ACKNOWLEDGEMENTS

We thank the anonymous reviewers for insightful comments.

7. REFERENCES

[1] D. Boneh and X. Boyen, “Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles," EUROCRYPT 2004, LNCS 3027, Springer-Verlag, pp. 223-238, 2004.

[2] M. Bellare, T. Kohno, and V. Shoup, “Stateful Public-Key Cryptosystems: How to Encrypt with One 160-bit Exponentiation," ACM CCS 2006. Full version available at http://eprint.iacr.org/2006/267.

[3] M. Bellare, C. Namprempre, “Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm," ASIACRYPT 2000, pp. 531-545, 2000.

[4] D. Boneh, M. Franklin, “Identity-Based Encryption from the Weil Pairing," SIAM J. of Compt., Vol. 32, No. 3, pp. 586-615, 2003.

[5] M. Bellare and P. Rogaway, “Random oracles are practical: a paradigm for designing efficient protocols," 1st ACM Conference on Computer and Communications Security, pp. 62-73, 1993.

[6] J.S. Coron, “On the exact security of Full Domain Hash", Proceedings of Crypto 2000, LNCS vol. 1880, Springer-Verlag, pp. 229-235, 2000.

[7] R. Canetti, S. Halevi, and J. Katz,“A Forward-Secure Public-Key Encryption Scheme," Eurocrypt 2003, LNCS Vol. 2656, Springer-Verlag, pp. 255-271, 2003.

[8] R. Cramer, V. Shoup,“Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack," SIAM J. of Comt., Vol. 33, pp.167-226, 2003.

[9] E. Fujisaki, T. Okamoto, “Secure Integration of Asymmetric and Symmetric Encryption Schemes", Proc. of Crypto ’99, LNCS 1666, Springer-Verlag, pp. 537–554, 1999.

[10] D. Galindo, “Boneh-Franklin Identity Based Encryption Revisited," In ICALP’05, LNCS 3580, pp.791-802, 2005. Full version available from the author’s homepage: http://www.cs.ru.nl/ dgalindo/ICALP2005full.pdf.

[11] K. Kurosawa and Y. Desmedt,“A New Paradigm of Hybrid Encryption Scheme," CRYPTO 2004, pp. 426-442, 2004.

[12] V. Shoup, “Using hash functions as a hedge against chosen ciphertext attack," Eurocrypt’00, pp. 275-288, 2000.

[13] V. Shoup, “Sequences of games: a tool for taming complexity in security proofs," manuscript, 2006. Available from http://www.shoup.net/papers/.

APPENDIX

A. PROOF OF LEMMA 1

Proof. Let A be an ind-sid-cca adversary against the scheme SIdent, we will build the adversary B, who controls the random oracle H_1 and plays with the game G as defined above. B runs A and simulates the environment for A as follows. It first gives A the security parameter κ and lets A output a target identity $ID^* \in \{0,1\}^n$. It then gives A the tuple $(q, G_1, G_2, \hat{e}, P, P_{pub}, n, k, H_1, H_2)$ as params, since it has $sp = (q, G_1, G_2, \hat{e}, P, H_2)$ and the public key $pk = (P_{pub}, Q)$, both of which received from the game G. (The value n, k are known from the description of H_1, H_2.) Note Q is not given to A and we will take advantage of this fact later. The adversary A makes many oracle queries, which are in turn handled by B, as follows:

– H_1-oracle query ID: If $ID = ID^*$, then B returns $Q_{ID^*} = Q$ to A. Otherwise, if ID was not queried before, B picks $b_{ID} \xleftarrow{\$} \mathbb{Z}_q$, and returns $Q_{ID} \leftarrow b_{ID}P$ to A. B also keeps a record of all asked (ID, Q_{ID}, b_{ID}) by a data structure H_1^{list} and uses this structure to answer identical repeated queries.

– Key extraction query ID $(\neq ID^*)$: B takes the corresponding (ID, Q_{ID}, b_{ID}) from H_1^{list}. (If such a record was not existed yet, B just goes ahead and calls the H_1 oracle algorithm so that the record exists. B can do so because it controls the random oracle H_1.) Then, B returns to A the value $b_{ID}P_{pub}(= b_{ID}(sP) = sQ_{ID})$, which is a legitimate private key for the identity ID .

– Encryption query m (only under ID^*): B takes m from A and gives the message to its own encryption oracle $SPub.Enc(sp, pk, \cdot, st)$ to receive a ciphertext of the form $(rP, SEnc_{K^*}(m))$, where K^* is internally computed as $H_2(Q, rP, \hat{e}(Q, P_{pub})^r)$ $(= H_2(Q_{ID^*}, rP, \hat{e}(Q_{ID^*}, rP_{pub})))$ by the encryption oracle. B then returns $(rP, SEnc_{K^*}(m))$ to A.

– (One) challenge query (m_0, m_1) (under ID^*): B gives (m_0, m_1) to its $LOR(sp, pk, \cdot, \cdot, b, st)$ oracle to obtain $(rP, SEnc_{K^*}(m_b))$ (K^* is as above). B then returns $C^* = (rP, SEnc_{K^*}(m_b))$ to A.

– Decryption query (ID^*, C), where $C \neq C^*$: B feeds C into its decryption oracle $SPub.Dec(sp, sk, \cdot)$, then takes and forwards the output to A. This decryption simulation is legitimate since the symmetric key in $SPub.Dec(sp, sk, \cdot)$ after feeding $C = (S, C_s)$ is $H_2(Q, S, \hat{e}(sQ, S))$, which is exactly the symmetric key to decrypt C under ID^* in SIdent.

Note that in the above simulation, the IBE's state is considered as (rP, rP_{pub}), while the PKE's state is (r, rP). The simulation is perfect from the viewpoint of the adversary A.

Finally, A returns a bit b', which is in turns output by B. From the fact that the simulation is perfect and the outputs are the same, we have

$$Adv^{ind-sid-cca}_{\text{SIdent}}(A) = Adv^{ind-cca}_{SPub}(B),$$

which concludes the proof of Lemma 1. □

B. PROOF OF LEMMA 2

Proof. We will prove the lemma via games G_0 and G_1 as defined as follows.

Let game G_0 be the original attack game, where B plays in game G as mentioned in Lemma 1. Recall that $K^* = H_2(Q, R, \hat{e}(Q, P_{pub})^r)$, where $R = rP$, is the (same) symmetric key in the oracle $SPub.Enc(sp, pk, \cdot, st)$ and the oracle $LOR(sp, pk, \cdot, \cdot, b, st)$. In addition, when B asks a decryption query $C = (S, C_s)$, game G_0 uses the key K^* to decrypt C_s if $S = R$; otherwise it uses $K = H_2(Q, S, \hat{e}(sQ, S))$. The change is just to rephrase game G_0, and does not affect the view of B.

Game G_1 is the same as G_0, except that the key K^* is now picked at random: $K^* \overset{\$}{\leftarrow} \{0,1\}^k$. Let F be the event that B queries H_2 at $(Q, R, \hat{e}(Q, P_{pub})^r)$ in game G_1. It is clear that games G_0 and G_1 are identical until F occurs. In particular, it follows that $T_0 \wedge \overline{F} = T_1 \wedge \overline{F}$ where T_0 and T_1 are the events the guess bit b' is equal to the challenge bit b in games G_0 and G_1, respectively. Thus, by the Difference Lemma [13],

$$|\Pr[T_0] - \Pr[T_1]| \quad \leq \quad \Pr[F]. \qquad (1)$$

We first claim that

$$|\Pr[T_1] - \frac{1}{2}| \quad = \quad Adv^{ind-cca}_{SE}(A_{SE}), \qquad (2)$$

$$\Pr[F] \quad = \quad Adv^{gap-bdh}_{\mathcal{G}}(A_{\mathcal{G}}). \qquad (3)$$

Note that $Adv^{ind-cca}_{SPub}(B) = |\Pr[T_0] - \frac{1}{2}|$ by definition, and Lemma 2 comes directly from (1), (2), and (3). It now suffices to justify the above claims.

We first justify (2). Intuitively, it comes from the fact that K^* is now truly random in game G_1 and hence $SEnc_{K^*}(m_b)$ gives no information about the challenge bit b if SE is IND-CCA secure. Formally, A_{SE} takes the parameter κ as input, uses B as a subroutine, and outputs what B outputs (which is the guess bit b'). Using κ, A_{SE} itself generates all the parameters for B and, in particular, it knows the encryption state (r, R) and the secret key sQ. A_{SE}, by definition, furthermore has access to oracles $SEnc(K^*, \cdot)$, $LOR(K^*, \cdot, \cdot, b)$, $SDec(K^*, \cdot)$ (K^* is suggestively used here.) Utilizing these oracles, A_{SE} returns $(R, SEnc(K^*, m))$, and $(R, LOR(K^*, m_0, m_1, b))$ respectively when B makes encryption query m and challenge query (m_0, m_1). When B asks a decryption query (S, C_s), A_{SE} returns $SDec_{K^*}(C_s)$ if $S = R$ and otherwise returns $SDec_K(C_s)$ where $K = H_2(Q, R, \hat{e}(sQ, S))$. Thus A_{SE} perfectly simulates the environment for B in game G_1, and forces B to guess the bit b in $LOR(K^*, \cdot, \cdot, b)$. The advantage of A_{SE} is $|\Pr[b' = b] - 1/2|$ by definition, which is exactly $|\Pr[T_1] - 1/2|$. Equation (2) follows.

We now justify (3), which is more challenging. We will construct $A_{\mathcal{G}}$ solving the Gap-BDH problem, making use of B as a subroutine. The input of $A_{\mathcal{G}}$ is an instance $(q, G_1, G_2, \hat{e})$ and random points (P, aP, bP, cP) and it will output $\hat{e}(P, P)^{abc}$ with the help of a DBDH oracle $\mathcal{O}_{dbdh}$, provided that the event F occurs. $A_{\mathcal{G}}$ runs B giving $sp = (q, G_1, G_2, \hat{e})$ to B as security parameter. It then puts $P_{pub} = aP$, $Q = bP$ and set $R = cP$. The public key is $pk = (P_{pub}, Q)$, and the state is considered as $st = (c, R)$ (Although $A_{\mathcal{G}}$ does not know a, b, c and $sk = abP$.) $A_{\mathcal{G}}$ also picks key $K^* \overset{\$}{\leftarrow} \{0,1\}^k$ and random bit $b \overset{\$}{\leftarrow} \{0,1\}$. Note that F is now the event that B queries $(bP, cP, \hat{e}(bP, aP)^c)$ to H_2. The way which $A_{\mathcal{G}}$ handles encryption queries m (to $SPub.Enc(sp,\ pk,\ \cdot,\ st)$) and challenge query (m_0, m_1) (to $LOR(sp, pk, \cdot, \cdot, b, st)$) is quite simple: it just returns $(R, SEnc_{K^*}(m))$ and $(R,\ SEnc_{K^*}\ (m_b))$ respectively to B.

Processing hash queries and decryption queries is however a bit intricate. Borrowing a technique originally used in [12], $A_{\mathcal{G}}$ manages the following lists, which is initially empty:

– $\mathcal{V}_1$: contains tuples of the form $(T, S, e, K) \in G_1^2 \times G_2 \times \{0,1\}^k$ for which K is the reply of the query (T, S, e) to H_2.

– $\mathcal{V}_2$: contains tuples of the form (T, S, e) for which $(T,\ S,\ e,\ K) \in \mathcal{V}_1$ for some $K \in \{0,1\}^k$ and $\mathcal{O}_{dbdh}(P, T, S, P_{pub}, e) = 1$ (namely, (P, T, S, P_{pub}, e) is a BDH tuple.)

– $\mathcal{V}_3$: contains tuples of the form (S, K) for which K is the reply of the query $(Q, S, \hat{e}(sk, S))$, where $sk = abP$, to H_2.

Intuitively, $A_{\mathcal{G}}$ uses $\mathcal{V}_1$, $\mathcal{V}_2$, $\mathcal{V}_3$ to handle both random oracle queries and decryption queries from B. The list $\mathcal{V}_1$ keeps records of all asked queries to H_2, and $\mathcal{V}_2$ is related to BDH tuples in $\mathcal{V}_1$. $\mathcal{V}_3$ is for processing decryption queries (S, C_s). Note that the symmetric key for decryption is $H_2(Q,\ S,\ \hat{e}(sk,\ S))$, and given S, the value $\hat{e}(sk, S)$ is fixed, although unknown to $A_{\mathcal{G}}$. Knowing the value S is enough for simulation, so it is kept in $\mathcal{V}_3$. Symmetric keys K may be added into both independent lists $\mathcal{V}_1$ and $\mathcal{V}_3$, so $A_{\mathcal{G}}$ must carefully check the both lists before returning new keys to B. $A_{\mathcal{G}}$ now processes hash and decryption queries from B as follows.

- Hash oracle query (T, S, e) to H_2: $A_{\mathcal{G}}$ checks if $(T,\ S,\ e,\ K) \in \mathcal{V}_1$ for some K. If so, K is returned to B. Otherwise, $A_{\mathcal{G}}$ uses the oracle $\mathcal{O}_{dbdh}$ to check whether (P, T, S, P_{pub}, e) is a BDH tuple.
  – If so: $A_{\mathcal{G}}$ first checks whether $T = Q(= bP)$ and $S = R(= cP)$. If this is the case (and hence the event F occurs), then $A_{\mathcal{G}}$ halts and outputs

e; otherwise it adds (T, S, e) to $\mathcal{V}_2$ and furthermore checks whether (S, K) is in $\mathcal{V}_3$ for some K. If so, K is returned to B and (T, S, e, K) is added to $\mathcal{V}_1$; otherwise a random key $K \in \{0,1\}^k$ is chosen, and (T, S, e, K) is added to $\mathcal{V}_1$.

  - If not: a key $K \in \{0,1\}^k$ is chosen at random, and (T, S, e, K) is added to $\mathcal{V}_1$.

- Decryption query $C = (S, C_s)$ (to $SPub.Dec(sp, sk, \cdot)$): If $(Q, S, e) \in \mathcal{V}_2$ for some e, then $A_{\mathcal{G}}$ takes the corresponding $(Q, S, e, K) \in \mathcal{V}_1$, and returns $SDec_K(C_s)$ to B. Otherwise, $A_{\mathcal{G}}$ checks whether (S, K) is in $\mathcal{V}_3$ for some K; if so, then $SDec_K(C_s)$ is returned to B, and if that is not the case, then $A_{\mathcal{G}}$ chooses $K \xleftarrow{\$} \{0,1\}^k$ and returns $SDec_K(C_s)$ to B.

$A_{\mathcal{G}}$ thus perfectly mimics the environment for B. As seen in the simulation of H_2, whenever the event F occurs, $A_{\mathcal{G}}$ will output $e = \hat{e}(P, P)^{abc}$, which is the Gap-BDH solution. Equation (3) follows. □

C. PROOF OF THEOREM 2

Proof. Let A be an adversary attacking Π_H, and its has advantage $\epsilon' = \epsilon(q_H + q_e)(1 - \frac{q_{extr}+q_d}{2^n})^{-1}$ in the IND-ID-CCA game. We will build an adversary B using A as a subroutine, and B's advantage in the IND-sID-CCA game is ϵ. B first chooses an index $j \xleftarrow{\$} \{1, \ldots, q_H + q_{extr} + q_e + q_d\}$ and sets a counter $ctr \leftarrow 0$. Since B is in the selective-ID game, it chooses $id^* \xleftarrow{\$} \{0,1\}^n$ and outputs the string as the “target" identity. B then receives params from the game, and feeds this parameter to A. B simulates the environment for A as follows.

– Hash oracle query $ID \in \{0,1\}^*$: We assume wlog that A never repeats a hash query and if A makes the other queries (below) related to an identity, it has already made a corresponding hash query. B first sets $ctr \leftarrow ctr + 1$ and puts $ID_{ctr} = ID$. If $ctr = j$, then B returns id^*; otherwise, it returns $id \xleftarrow{\$} \{0,1\}^n$.

– Key extraction query ID: We know that $ID = ID_i$ and $H(ID) = id_i \in \{0,1\}^n$ for some $1 \leq i \leq q_H + q_{extr} + q_e + q_d$. If $id_i \neq id^*$ then B forwards id_i to its own key extraction oracle, obtains back the private key corresponding to id_i, and gives the key to A. However, in the unlikely event that $id_i = id^*$, B cannot respond to this query, so it halts and outputs a random bit as a guess of the challenge bit b of the game.

– Encryption query (ID, m): B forwards $(H(ID), m)$ to its own encryption oracle, gets back the ciphertext and returns it to A.

– Challenge query (ID^*, m_0, m_1): We know that $ID^* = ID_k$ and $H(ID^*) = id_k \in \{0,1\}^n$ for some $1 \leq k \leq q_H + q_{extr} + q_e + q_d$. If $k = j$ (and hence $id_k = id^*$), B forwards (m_0, m_1) to its own challenger, receives back the challenge ciphertext C^* and gives it to A. Otherwise, namely $k \neq j$, B halts and outputs a random bit.

– Decryption query $(ID, C)(\neq (ID^*, C^*))$: We know that $ID = ID_k$ and $H(ID) = id_k \in \{0,1\}^n$ for some $1 \leq k \leq q_H + q_{extr} + q_e + q_d$. If $id_k \neq id^*$ or $C \neq C^*$ then B forwards (id_k, C) to its own decryption oracle, gets back the result and gives it to A. Otherwise, B halts and outputs a random bit.

Finally, if B does not halt as above, it outputs what A outputs. This ends the description of B.

We begin to analyze B's advantage in guessing the challenge bit b. Denote b' the output of B, which is either a random bit or A's output. We call NoHalt the event that B does not halt before A's output. It is clear that if the event NoHalt occurs, then B perfectly simulates the environment for A, and the output of B is exactly the output of A, so that $|\Pr[b' = b|\text{NoHalt}]| - 1/2 = \epsilon'$. On the other hand, if the event $\overline{\text{NoHalt}}$ occurs, then the output of B is random, so that we have $|\Pr[b' = b|\overline{\text{NoHalt}}]| = 1/2$. Thus,

$$\begin{aligned} & Adv_{\Pi}^{ind-sid-cca}(B) = |\Pr[b' = b] - \frac{1}{2}| \\ = \quad & |\Pr[b' = b|\text{NoHalt}]\Pr[\text{NoHalt}] \\ + \quad & \Pr[b' = b|\overline{\text{NoHalt}}]\Pr[\overline{\text{NoHalt}}] - \frac{1}{2}| \\ = \quad & |(\epsilon' + \frac{1}{2})\Pr[\text{NoHalt}] + \frac{1}{2}(1 - \Pr[\text{NoHalt}]) - \frac{1}{2}| \\ = \quad & \epsilon' \Pr[\text{NoHalt}]. \end{aligned}$$

We now estimate Pr[NoHalt]. Additionally denote NoHalt$_1$ the event that B does not halt while processing the challenge query, and NoHalt$_2$ the events that B does not halt while processing the other queries. We have $\text{NoHalt} = \text{NoHalt}_1 \cap \text{NoHalt}_2$ and hence

$$\Pr[\text{NoHalt}] = \Pr[\text{NoHalt}_2]\Pr[\text{NoHalt}_1|\text{NoHalt}_2].$$

We now proceed to compute $\Pr[\text{NoHalt}_1|\text{NoHalt}_2]$. Note that $\Pr[\text{NoHalt}_1|\text{NoHalt}_2] = \Pr[k = j|\text{NoHalt}_2]$. The conditional event NoHalt$_2$ implies that $k \neq j$ while B processes extraction and decryption queries, so that

$$\begin{aligned} \Pr[k = j|\text{NoHalt}_2] &= \frac{1}{(q_H + q_{extr} + q_e + q_d) - (q_{extr} + q_d)} \\ &= \frac{1}{q_H + q_e}. \end{aligned}$$

Thus

$$\Pr[\text{NoHalt}_1|\text{NoHalt}_2] = \frac{1}{q_H + q_e}.$$

We now compute $\Pr[\text{NoHalt}_2]$. The event NoHalt$_2$ means B does not halt while processing extraction queries and decryption queries, which in turn means that $id_i \neq id^*$ for all id_i appeared in the queries. Since id^* is chosen randomly from $\{0,1\}^n$, we have $\Pr[\text{NoHalt}_2] = 1 - \frac{q_{extr}+q_d}{2^n}$.

Thus, $\Pr[\text{NoHalt}] = \frac{1}{q_H+q_e}(1 - \frac{q_{extr}+q_d}{2^n})$, and hence

$$Adv_{\Pi}^{ind-sid-cca}(B) = \epsilon' \Pr[\text{NoHalt}] = \epsilon,$$

which concludes the proof. □